(54) Title: APPARATUS AND METHOD TO MINIMIZE INCOMING DATA LOSS
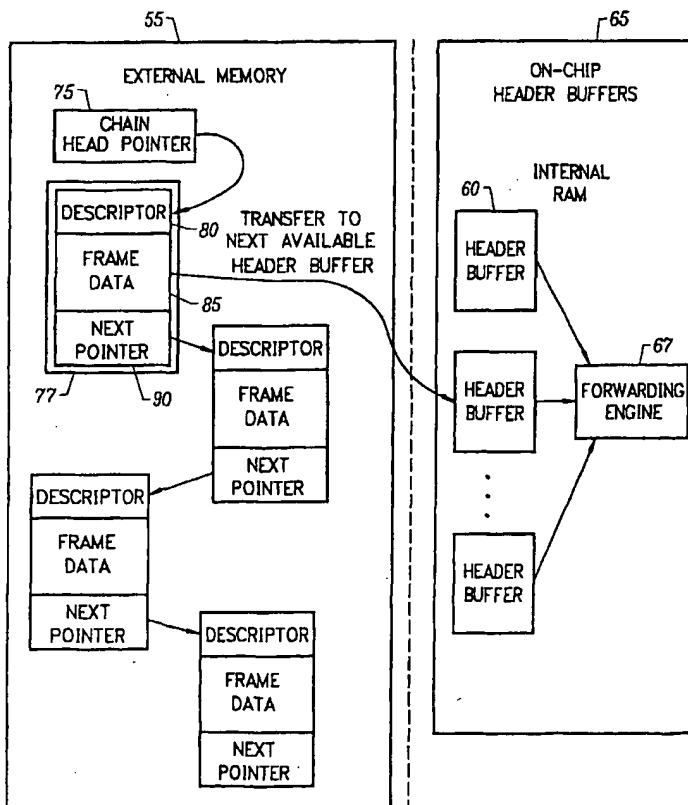
(57) Abstract: As computer network switches provide additional capabilities, including level 3 and level 4 management and data transfers, the hardware foundations that they rely on are overwhelmed by the ever increasing amount of data they must route. As a switching unit receives data frames, a forwarding process is used to route the frames to one of a plurality of output channels. Higher levels of protocol services require additional computational resources. With increasing data volume, the forwarding process can not be completed in real-time. With the failure of the forwarding process, data frames can not be routed and are dropped from the stream. A predominate factor in the number of frames dropped per unit time is the availability of header buffers. The present invention monitors the availability of header buffers. When header buffers are not available, incoming data frames are stored in an external memory. As header buffers are freed up by the forwarding process, special hardware moves the header portion of the data frames stored in the external memory into an available header buffer.

WO 01/05123 A1

Apparatus and Method to Minimize Incoming Data Loss

BACKGROUND OF THE INVENTION

TECHNICAL FIELD

The invention relates to the processing and management of data flowing through a computer network switch. More particularly, the invention relates to an apparatus and method for minimizing incoming data loss in output queuing.

DESCRIPTION OF THE PRIOR ART

Computer networks are constructed by tying together a plurality of switching units. The switching units receive data from various sources in a quantum known as frame. As computer networks continue to proliferate throughout the world, switching units must be able to route an ever-increasing bandwidth of data. As the bandwidth increases, switching units must be able to handle a greater number of data frames per given unit of time.

The switching units themselves rely on various strategies to ensure that the total frame rate can be accommodated. In the previous art, as frame rate increased the hardware foundation of the switching unit would be taxed to such an extent that some frames would inevitably be lost. These lost frames are known as dropped frames.

Computer networks built from these switching units could tolerate dropped frames by using higher-level transport protocols that guarantee the delivery of data. The technology used in switching units has traditionally provided only the lower levels of networking services. In terms of the ISO Stack, switching units would provide only Level 1 and Level 2 services. In the case of dropped frames, the computers using the network would be required to provide the higher level services of the ISO Stack in order to ensure delivery of the data frame.

Relying on the higher-level protocols to guarantee delivery is the traditional solution to the dropped frame problem, but this technique aggravates the need for bandwidth. As frames are dropped, the need for bandwidth

increases as the computers seeking to provide Level 3 and Level 4 protocol services retransmit the dropped frames. The higher aggregate bandwidth results in even more dropped frames as the switching units futilely attempt to route the additional data.

5

Recent advances have enabled the switching units to provide Level 3 and Level 4 protocol services directly. At first inference, this should have an ameliorating effect on overall bandwidth requirements, especially when attempts are made to retransmit dropped frames. However, these higher-

10   level protocol services further tax the already challenged hardware foundations of the switching units. The net and frustrating result of this is that even more frames are dropped.

In yet another implementation, Level 3 and Level 4 protocols are used to

15   provide a feature called web cache redirection. A web cache may be located off one port of the switch while an actual web page that is being cached may exist off a different port. If a frame containing a request for the given web page comes into the switch, a layer-2 switch would always forward the request to the port where the actual web page is located, it does not

20   understand the concept of a web cache and never looks deep enough into the frame to determine that it is a web page request. A properly configured layer-4 switch would route the frame to the web cache. The web cache is generally located closer to the requestors than the actual web pages within the global network, which results in an overall reduction in network traffic and

25   faster response time to the client.

Another example of the prior art utilizing layer 4 protocols is a feature called server load balancing. The switch is configured for client ports and server ports. The clients make requests for a given service that may be satisfied b y

30   any one of a group of physical servers. The switch balances the requests across the servers according to an algorithm selected by the user. Examples include round robin and least connections. This layer-4 feature improves response time to the clients by making more efficient use of a bank of servers. It does not, however, reduce overall network traffic.

35

Relegating the Level 3 and Level 4 services embodied in these types of features to the switching units is highly desirous because the amount of data traffic in the overall network can be reduced. This result is achieved because dropped frames can be handled directly between the two switching units that

2

are effecting the data connection rather than between two client computers and the expanse of the entire network.

5

SUMMARY OF THE INVENTION

The methods and apparatus described herein implement a novel and unique facility that provides for the management of data frames arriving at a switching unit. The switching unit uses header buffers to coordinate the routing of data frames to the proper output queue. Unlike prior art, the new switching unit creates data structures in external memory and stores newly arriving data frames in the external memory whenever there are no header buffers available. As header buffers are freed, the new switching unit reads the header portion of the data frames stored in external memory and writes the header portion into an available header buffer.

20

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram that depicts a typical network topology and is used to place the present invention into context;

Fig. 2 is a data flow diagram that depicts the data flow of data frames flowing through a switch unit;

Fig. 3 is a data structure diagram that illustrates the method used to create expanded storage for data frames when header buffers are not available;

Fig. 4 is a functional block diagram that presents a block diagram of a queue switch according to the invention.

35

DETAILED DESCRIPTION OF THE INVENTION

The invention is a method and an apparatus that minimizes the loss of data flowing through a switching unit. The apparatus is preferably embodied as a queue switch.

5   *Network Topology*

It is first necessary to understand the application of switching units as it pertains to the loss of data to appreciate fully the utility of the invention

10   Fig. 1 depicts the interconnection of a plurality of switching units 5 through the use of communication ports 10 from each switching unit to other switching units. By connecting these switching units to each other in this manner, a wide area network 15 is realized. Other clusters of switching units, or lower capability hubs and routers can be interconnected to create local area

15   networks 20. Attached to these local area networks 20 are a plurality of network clients. For the purposes of the illustration here, a sender 25 and a receiver 30 constitute two of a vast potential of network clients that can be connected to disperse local area networks that are then connected together by the wide area network.

20

As can be inferred from Fig. 1, the sender 25 can send frames of data to any other client connected to any appendage of the local or extended network. The switching units 5 play a pivotal role in routing the data frames dispatched by the sender 25. The function provided by the switching units 5 is to

25   determine, on a frame by frame basis, where to direct the frame so as to ensure delivery to the receiver 30.

As can also be inferred from Fig. 1, the switching units 5 normally receive data from the sender 25 on one physical port, however this does not need to

30   be the case. What is also apparent is that each switching unit 5 has a plurality of ports, each capable of receiving inputs of data frames.

Forwarding Process

35   Fig. 2 depicts how data frames flow through each switching unit 5. Internal to a switching unit 5, data frames arrive via a plurality of input ports. Each input port is serviced by a corresponding first-in-first-out (FIFO) buffer 35. The input FIFO's 35 receive data frames from external network connections and

provide the elasticity needed to assimilate the aggregate data throughput arriving at the port.

5   In switching units presently known, a forwarding process is executed that examines each data frame and determines to which output queue 45 it is to be directed. Once the data frame is directed to an output queue 45, it is scheduled to be transmitted to the next switching unit, or router in the case of a connection to a local area network.

10   The source of routing information that the forwarding process 40 uses to determine the queue each data frame must be delivered is stored in tabular databases. The forwarding process 40 creates and maintains a dynamic map, referred to herein as a forwarding database 50. The forwarding process 40 derives knowledge about the entire network, both the wide area and the
15   plurality of local networks, as it seeks to optimize connection and routing tables. This activity requires significant processing resources to achieve. Another source of routing information is a static map that is referred to herein as a switch configuration database 55. The switch configuration 55 is developed through *apriori* knowledge of the network configuration and is
20   entered directly into the switching unit 5.

The forwarding process 40 uses sophisticated algorithms to learn about the topology of the extended network to maintain the forwarding data base 50. All layer-2 switches have a learning algorithm for layer-2 network addresses,
25   and maintain a layer-2 forwarding database. Switches that perform layer-3 and layer-4 functions maintain extra tables that track TCP/IP sessions and include client and server information.

When a frame arrives from the input FIFO of a layer-2 switch, the switch
30   searches the database for the given destination address. If successful, the search returns the output port for the frame and the switch places the frame on the associated output queue. If unsuccessful the switch floods the frame to all ports except the one it came in on. Prior art includes switches that perform this simple forwarding algorithm at full wire speed.
35
When a frame arrives from the input FIFO of a layer-4 switch, the forwarding engine has a much larger job. If a client frame indicates the start of a new TCP/IP session, the switch must choose an appropriate server and augment the layer-4 database to include the new session before placing the frame

onto the selected output queue. The server selection is based on the switch configuration and in some cases server loading information also kept by the switch. Both web cache redirection and server load balancing, discussed above, require this type of processing.

5

Hence, as each frame arrives from the input FIFO 35, the forwarding process 40 uses these complex algorithms to determine to which output queue each frame must be directed. Among other things, these algorithms enable the forwarding process 40 to anticipate the load on network servers that source
10   the data frames and to track the utilization of communication channels that interconnect the plurality of switching units and routers that form the extended network. These algorithms work collectively to make the extended network as efficient as possible.

15   The forwarding algorithms have become so complex that the forwarding process 40 can not be executed for each arriving frame in real time. As a result, data frames can not be routed and are consequently dropped.

Header Buffers and Data Overflow

20
Incoming data frames are split into a header and a payload. The header contains most of the information required to forward the frame. The payload contains the remainder of the frame. Prior art switching units rely on a series of header buffers that temporarily store incoming frame headers while the
25   forwarding process executes. These header buffers receive data frames from a plurality of input FIFOs 35 and present the data frames to the forwarding process 40. The availability of header buffers is limited and their parallel and high-speed nature precludes mass replication of this resource. As the rate of incoming data frames peaks, the forwarding process 40 can not
30   free header buffers fast enough to accommodate newly arriving data frames. As a result, data frames are lost, _i.e._ dropped.

Fig. 3 illustrates one of the main novelties of the invention. To preclude, or at least minimize the loss of data frames when header buffers are not available,
35   the present switching unit provides for storage of data frames in an external memory element 55 and the ability to read the frame headers back when header buffers eventually become free. The plurality of header buffers 60 are actually internal to an application specific integrated circuit (ASIC) referred

to herein as a network engine 65. The network engine 65 further comprises a forwarding engine 67 that performs the forwarding process 40.

5   Specialized hardware, embodied in the network engine 65, manages the creation of data structures in the external memory 55. Each time a data frame arrives, it is forwarded to one of the plurality of header buffers 60. In the preferred embodiment of the invention, a data structure is also created in the external memory 55 and the incoming data frame stored therein as well. This is optional. What is important, though, is that when a header buffer is not

10   available, the external data structure must be created and the incoming data frame must be stored therein to preclude its loss.

The specialized hardware manages data structures in a traditional chained link-list form. When the first data structure is created it is identified by it's address

15   and that address is stored in an overflow FIFO head-pointer referred to herein a chain head-pointer 75. Each data structure 77 is comprised of three major fields: a queue descriptor 80, a frame data 85, and a next data structure pointer 90. The queue descriptor 80 is used in identifying and managing the routing of the data frame. Frame data 85 are used to store the data from the

20   incoming data frame. The next data structure pointer 90 is used to identify the next data structure in the chain. The next data structure pointer of the last data structure in the chain is set to a null value as an indication that that particular data structure is the last in the chain.

25   The specialized hardware that manages the creation of new data structures in the external memory 55 also monitors the availability of header buffers 60. When a header buffer becomes available, the specialized hardware transfers the frame data from the first data structure in the chain to that buffer. It does so by using a chain head pointer 75 to identify the first data structure in the chain

30   and then it transfers the contents of the header portion of the frame data field 85 from the first data structure to the available header buffer 60. The address of the first data structure is then transferred to a register associated with the header buffer that is used to store the frame data. This address is later used to link the frame data in the external memory 55 with an output queue. Once

35   the data are transferred, the specialized hardware reads the value of next data structure pointer 90 from the first data structure and stores that value in the chain head-pointer 75. This makes the next data structure, pointed to by the next pointer, the first data structure in the chain.

The data structure that is used as the source of data for an available header buffer is not immediately destroyed. It is retained in the external memory and the address of this data structure is then used to point to the frame data when it is assigned to an output queue.

Hardware Embodiment

Fig. 4 depicts the preferred embodiment of the method of the invention implemented as a hardware apparatus. A network engine 65 and the external memory 55 collectively comprise an output queuing switch 135. The network engine 65 comprises a network receiver circuit 100, a wire input FIFO 105, a forwarding engine 110, a queue linker 120, a switch queue 125, a switch media access controller (MAC) 130, and a memory arbiter 115. The external memory 55 is preferably a high-speed random access memory, such as a synchronous dynamic random access memory (SDRAM).

In the preferred embodiment of the invention, the network receive circuit 100 interfaces with the actual communication channel that accepts data frames from an external source. This could be an Ethernet connection, a fiber optic receiver, or a modem or the like. The network receive circuit 100 accepts the data frames and presents them to a wire-input FIFO.

The wire input FIFO 105 receives data frames from the network receive circuit 100 and immediately attempts to transfer the data to a header buffer. Header buffers are depicted in Fig. 4 as a header buffer stack 106.

In the preferred embodiment of the invention, the wire input FIFO 105 comprises the specialized hardware that manages the creation of data structures in external memory as discussed above. In the event that a header buffer is not available, the wire input FIFO 105 creates a data structure in the external memory 55 according to the method of the invention disclosed herein. Once the data structure is created, the data frame is stored in the data structure until a header buffer becomes available. Once a header buffer becomes available, wire input FIFO 105 reads the first data frame from external memory 55 and transfers it to the available header buffer.

The forwarding engine 110 reads the information in the data frame, including source and destination addresses and IP protocol data that it needs to forward the data frame. Results of the forwarding process are delivered by the

forwarding engine 110 back to the wire input FIFO 105. The wire input FIFO 105 sends the pointer to the data frame, located in external memory, to the queue linker 120. The queue linker 120 reads a descriptor 80 to determine to which output queue the data frame is to be routed and initiates the switch

5   queue 125. The queue linker 120 uses information in the descriptor 80 to determine which output queue to link the frame to. The queue linker 120 performs the link operation and notifies the switch queue 125 by means of an initiation signal that the output queue is no longer empty.

10   After receiving an initiation signal from the queue linker 120, the switch queue 125 de-links frames from non-empty output queues and retrieves the data frame from the external memory 55. The switch queue 125 then directs the data frame to the switch media access controller 130. The switch media access controller 130 then forwards the data frame to one of a plurality of

·15   output FIFOs that it maintains and manages. The switch media access controller 130 then delivers the queues to a switch port that puts the data frames, now queued in a plurality of output FIFOs, out onto a communication channel.

20

Although the present invention is described herein with reference to the preferred embodiment, one skilled in the art will readily appreciate that other applications may be substituted for those set forth herein without departing from the spirit and scope of the present invention. Accordingly, the present

25   invention should only be limited by the Claims included below.

CLAIMS

1. A method for reducing the loss of data in an output queuing switch comprising the steps of:

    receiving data frames from a network receiver circuit;

    transferring said data frames to a working register as long as working registers are available;

    creating a data structure in a memory when there are no longer any working registers available;

    storing a data frame in said data structure when it is received from said network receiver circuit;

    monitoring availability of working registers;

    transferring a data frame from a data structure to a working register when a working register becomes available;

    notifying a forwarding process that a new data frame has been transferred to an available working register; and

    assigning said new data frame to an output queue based on a notification received from said forwarding process.

2. The method of Claim 1, further comprising the step of creating a chain head pointer.

3. The method of Claim 2, wherein said data structure further comprises a pointer that points to a next data structure to form a chained list of data structures; and wherein said step of creating a data structure in memory further comprises the step of setting said chain head pointer to identify said new data structure if there are no active data structures or setting said next data structure pointer of a previously created data structure to identify said new data structure otherwise.

4. The method of Claim 3, wherein the source data structure used in said step of transferring a data frame from a data structure to a working register is identified by a chain head pointer and said step itself further comprises the step of updating said chain head pointer to identify the data structure identified in said source data structure's next-data-structure pointer.

5. A method for reducing the loss of data in an output queuing switch, comprising the steps of:

10

receiving data frames from a network receiver circuit;

transferring the data frames to a working register so long as working registers are available;

creating a data structure in memory and storing a data frame in the data structure when it is received from the network receiver circuit;

monitoring the availability of working registers;

transferring a data frame from a data structure to a working register when the working register becomes available;

notifying a forwarding process that a new data frame has been transferred to an available working register;

assigning the data frame to an output queue based on a notification received from the forwarding process.

6.   The method of Claim 5, further comprises the step of creating a chain head pointer.

7.   The method of Claim 6, wherein said data structure further comprises a pointer that points to the next data structure in order to form a chained list of data structures and the step of creating a data structure in memory further comprises the step of setting the chain head pointer to identify the newly created data structure if there are no active data structures or setting the next data structure pointer of the previously created data structure to identify the newly created data structure.

8.   The method of Claim 7, wherein the source data structure used in the step of transferring a data frame from a data structure to a working register is identified by a chain head pointer and the step itself further comprises the step of updating the chain head pointer to identify the data structure identified in the source data structure's next-data-structure pointer.

9.   An output queuing switch comprising:

network receiver circuit that accepts data from a network;

network engine having a plurality of header buffer registers;

memory element;

wire input first-in-first-out buffer that::

    accepts data frames from said network receiver circuit;

writes the data frames into said buffer registers so long as there is an available buffer register;

writes the data frames into said memory element when there are no header buffer registers available;

tracks the availability of header buffer registers; and

retrieves data frames from said memory element and stores them into header buffer register as they become available;

forwarding engine that reads the contents of said header buffer registers, determines the appropriate output queue and conveys the appropriate output queue to said wire input first-in-first-out buffer;

queue linker that accepts queue assignments from said wire input first-in-first-out buffer and builds queues comprised of chained data structures;

switch manager that retrieves queue data from said memory; and

switch media access controller that accepts queue data from said switch manager and dispatches the queue data to a network port.

10.   The output queuing switch of Claim 10, wherein the wire input first-in-first-out buffer further comprises:

overflow chain head-pointer that indicates the oldest data structure in a chain;

chain data structure manager that creates data structures comprising a next-element-pointer and a data frame record to receive a data frame whenever a data frame is written into the memory element and updates said next-element-pointer to identify the data structure created immediately subsequent the first data structure if there are other active data structures in memory or updates the overflow chain head-pointer to indicate the first active data structure if the newly created data structure is the only active data structure; and

transfer tracker that updates the overflow chain head-pointer to indicate the data structure indicated by the next-element-pointer in a data structure that is transferred to an available header register.

11.   An output queuing switch comprising:

network receiver circuit that accepts data from a network;

network engine having a plurality of header buffer registers;

memory element;

wire input first-in-first-out buffer that::

accepts data frames from said network receiver circuit;

12

writes the data frames into said buffer registers so long as there is an available buffer register;

writes the data frames into said memory element;

tracks the availability of header buffer registers; and

retrieves data frames from said memory element and stores them into header buffer register as they become available;

forwarding engine that reads the contents of said header buffer registers, determines the appropriate output queue and conveys the appropriate output queue to said wire input first-in-first-out buffer;

queue linker that accepts queue assignments from said wire input first-in-first-out buffer and builds queues comprised of chained data structures;

switch manager that retrieves queue data from said memory; and

switch media access controller that accepts queue data from said switch manager and dispatches the queue data to a network port.

12.   The output queuing switch of Claim 11, wherein the wire input first-in-first-out buffer further comprises:

overflow chain head-pointer that indicates the oldest data structure in a chain;

chain data structure manager that::

creates data structures comprising a next-element-pointer and a data frame record to receive a data frame whenever a data frame is written into the memory element

updates said next-element-pointer to identify the data structure created immediately subsequent the first data structure if there are other active data structures in memory or updates the overflow chain head-pointer to indicate the first active data structure if the newly created data structure is the only active data structure; and

transfer tracker that updates the overflow chain head-pointer to indicate the data structure indicated by the next-element-pointer in a data structure that is transferred to an available header register.
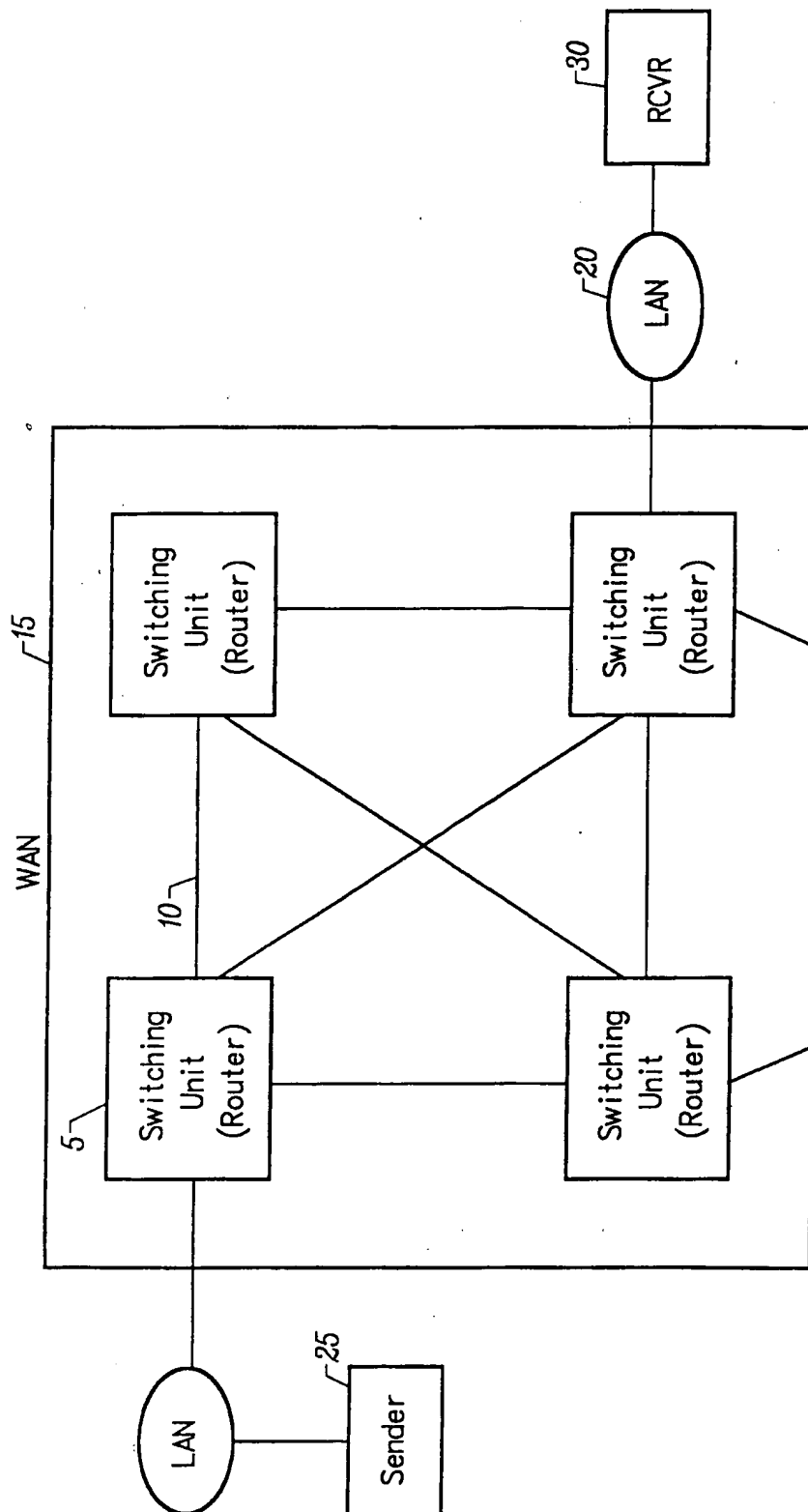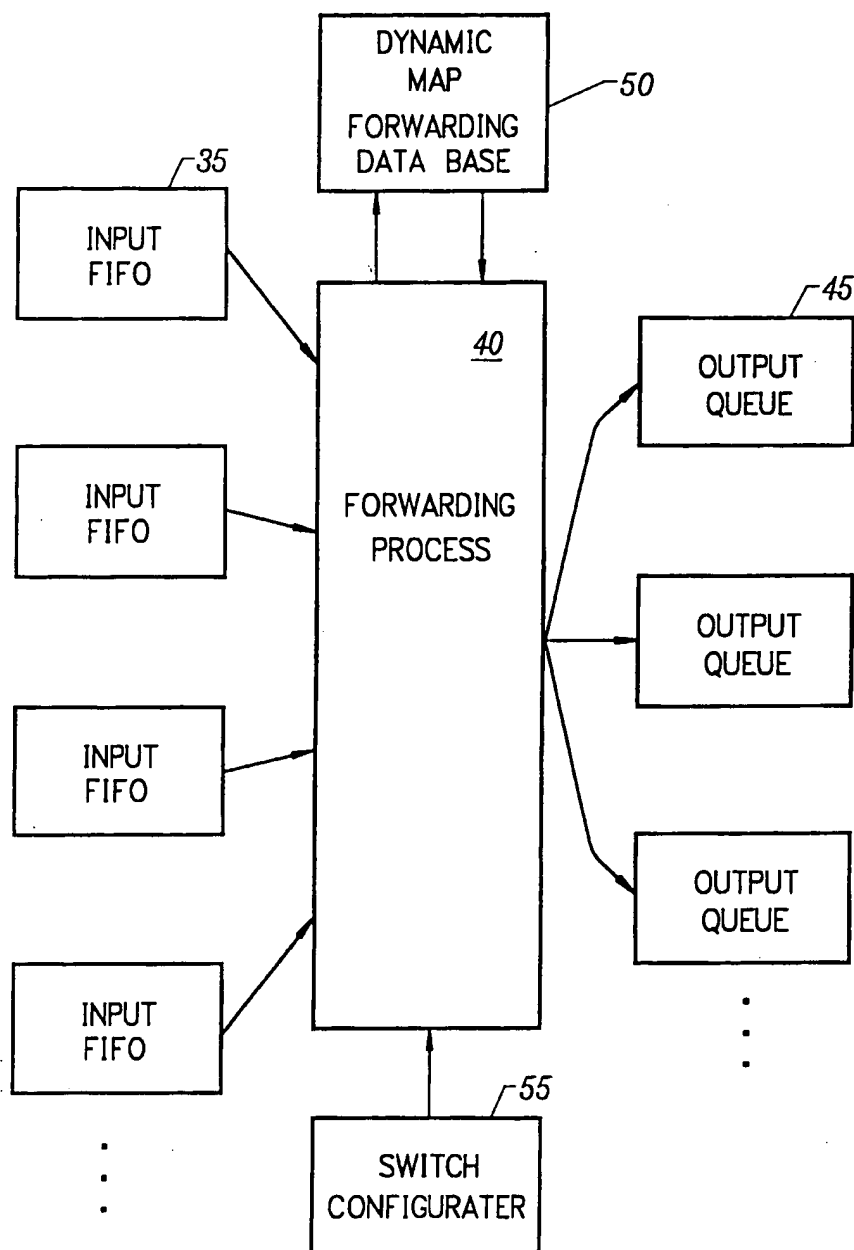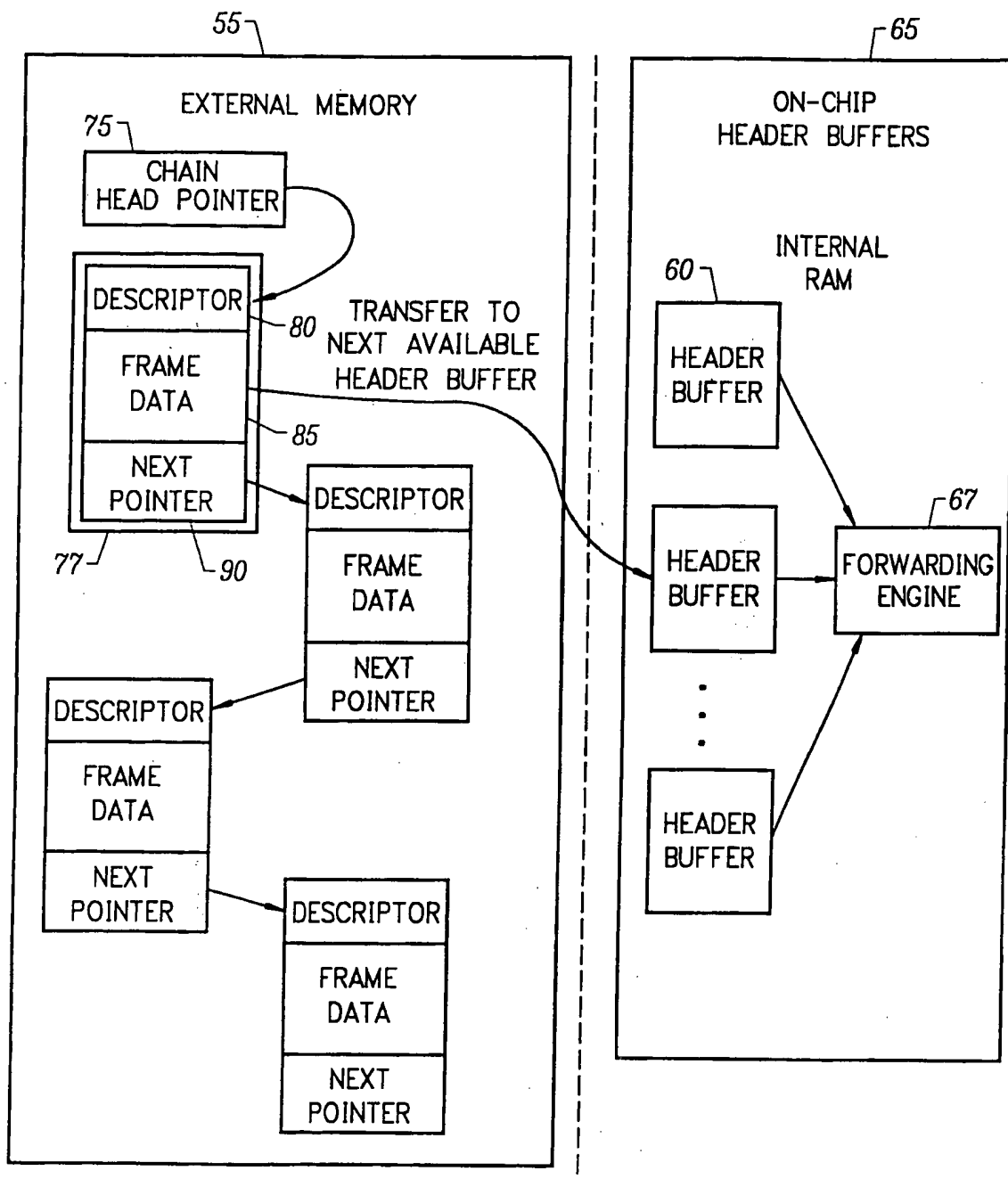
1/4

FIG. 1

FIG. 2

3/4

FIG. 3

4/4

RX SIDE OF GNE   ─135

ENET
RX

100 ─┐   105 ─┐   106 ─┐   HBR's
        RX   WIRE   HEAD BUFFER   FORWARDING   ─65
        EMAC   INPUT   STACK   ENGINE
                    FIFO   ─110
                            RESULTS

120 ─┐
        QUEUE
        LINKER

125 ─┐
        SWITCH   MEMORY   ─55
        QUEUE   ARBITER
                                                               EXTERNAL
                                                               MEMORY
                                            ─115

    ─130
        SWITCH
        MAC

SWITCH
PORT

FIG. 4

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7   H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7   H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the International search (name of data base and, where practical, search terms used)

EPO-Internal, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	RUTSCHE E: "THE ARCHITECTURE OF A GB/S MULTIMEDIA PROTOCOL ADAPTER" COMPUTER COMMUNICATIONS REVIEW,US,ASSOCIATION FOR COMPUTING MACHINERY. NEW YORK, vol. 23, no. 3, 1 July 1993 (1993-07-01), pages 59-68, XP000412143 ISSN: 0146-4833 figure 1 page 62, line 7-29	1,5,9,11
A	US 5 398 245 A (HARRIMAN JR EDWARD S) 14 March 1995 (1995-03-14) abstract	1,5,9,11
A	EP 0 853 404 A (DIGITAL VISION LAB CORP) 15 July 1998 (1998-07-15) abstract	1,5,9,11

-/--

| X | Further documents are listed in the continuation of box C. | | X | Patent family members are listed in annex. |

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
9 November 2000	16/11/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Dhondt, E

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT		
Category *	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 634 015 A (MELEIS HANAFY E-S ET AL) 27 May 1997 (1997-05-27) column 9, line 25-30 column 10, line 25-35 ―――	1,5,9,11

1

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
US 5398245	A	14-03-1995	AU	2756392 A	03-05-1993
			DE	69228758 D	29-04-1999
			DE	69228758 T	02-12-1999
			EP	0606368 A	20-07-1994
			JP	7501189 T	02-02-1995
			WO	9307692 A	15-04-1993
EP 0853404	A	15-07-1998	JP	10200574 A	31-07-1998
			US	5809078 A	15-09-1998
US 5634015	A	27-05-1997	US	5367643 A	22-11-1994
			BR	9200413 A	13-10-1992
			EP	0498201 A	12-08-1992
			JP	2518986 B	31-07-1996
			JP	4336729 A	24-11-1992